Embedded Evolutionary Robotics: The (1+1)-Restart-Online Adaptation Algorithm

Jean-Marc Montanier, Nicolas Bredeche

Embedded Evolutionary Robotics: The (1+1)-Restart-Online Adaptation Algorithm

Jean-Marc Montanier
TAO - Univ. Paris-Sud, INRIA, CNRS
LRI, Bat. 490, 91405 Orsay, France
montanier@lri.fr

Nicolas Bredeche
TAO - Univ. Paris-Sud, INRIA, CNRS
LRI, Bat. 490, 91405 Orsay, France
nicolas.bredeche@lri.fr

ABSTRACT

This paper deals with online onboard behavior optimization for an autonomous mobile robot for autonomous online adaptation in an unknown environment. The work presented here extends the (1+1)-online algorithm, which was introduced in [3]. This algorithm is a variation of a famous Evolution Strategies [18] adapted to autonomous robots. In this paper, we address a limitation of this algorithm regarding the ability to perform global search whenever a local optimum is reached. A new implementation of the algorithm, termed (1+1)-restart-online algorithm, is described and implemented within the Symbrion robotic Cortex M3 microcontroller as well as on a real mobile robot. Results from the experiments show that the new algorithm is able to escape local optima and, as a consequence, converge faster and provides a richer set of relevant controllers.

1. INTRODUCTION

Let's imagine an autonomous mobile robot tailored for exploration that could be dropped in a wide variety of unknown environments, from a dense tropical forest to an exhausted gold mine abandoned 100 years ago. Even before starting to explore its environment, this kind of robot would need to be able to adapt to its immediate surrounding, that is figuring out what shape and/or what behavior is most fitted to sustain its energy level. In this setup, the robot control architecture would be preliminary driven by the specific, unpredictable, properties of the environment.

This paper focuses on such a problem, that is the design of a control architecture for an autonomous mobile robot in an unknown environment. To do so, there exists a wide variety of approaches depending on the problem to be solved, from hand-crafted reactive behavior [2] to optimal control approaches [7]. In the aforementionned situation however, it is difficult, if not impossible, to a priori specify the environment and the task at hand, which implies that most of the existing approaches are not fitted. This is a typical problem in Robotics that may be addressed with learning and optimization[20, 8]. Moreover, we address the problem where little is known about the objective function. This means that the task is poorly described as a single efficiency measure (e.g. minimize energy consumption, maximize exploration, etc.), which is often delayed and noisy. In this scope, Evolutionary Robotics provides optimization algorithms based on Evolutionary Algorithms which are fitted to this class of problems.

Evolutionary Robotics [15, 7] ("ER") takes inspiration from nature by combining two partly antagonist mechanisms. On the one hand, *selection* of the most fitted individuals tends to ensure convergence of the algorithm. On the other hand, *variation* over the properties of selected individuals through mutation and recombination tends to provide new original solutions. The general framework of these algorithms, termed Evolutionary Algorithms ("EA"), is often referred to as stochastic population-based optimization algorithms and has been applied to a wide variety of problems [4].

In Autonomous Robotics, EA is often used as an optimizer for Artificial Neural Network architectures to control autonomous robots in a wide variety of control task, from navigation and non-linear control to swarm coordination and cooperation (see [8] for examples of applications). The quality, or *fitness* of a given genotype is computed by creating a *phenotype* (an artificial neural network for robot control with optimized weights) and *evaluated* it in the environment (assessing the performance of the resulting robot behavior). Based on this evaluation methodology, a *population* of *genotypes* is evaluated, from which the (usually) better genotypes are selected and go through a variation process so as to renew the population. This process is then iterated until a termination criterion is matched (e.g. maximum number of evaluation, performance, etc.) and is usually referred to as *off-line ER*. While off-line ER can be used to address non-linear control problems or poorly defined objective function, it fails to provide a continuous autonomous optimization process as control over the initial condition for genome evaluation is required, which often imply either costly human interventation or the use of simulation [12]. Moreover, evaluation of a genome requires reliability, ie. the fact that one evaluation session must be relevant with regards to the problem at hand (ie. concept drift is not addressed).

Embodied ER, introduced in [6], is a sub-field of ER that precisely addresses the problem of changing environments without constant human manutention. In this setup, the Evolutionary Algorithm runs within the robot (or group of robots), acting as an embedded optimization algorithm. Embodied is defined as both online (the adaptation/learning process never stops) and onboard (the optimization algorithm and evaluation process are part of the control loop). To date, only few, but promising, works have adressed this topic [24, 14, 22, 23, 9, 13, 5, 25, 16, 11, 19]. Despite strong advantages regarding continuous adaptation and autonomy with regards to a human supervisor, running an embedded EA within a single robot also emphasizes some specific issues :

– Unknown fitness landscape : the typical fitness landscape in ER is both multi-modal (many local minima) and partly neutral (many close genotype give perform in a similar way). One reliable assumption is that of strong causality[17], the fact that small variations in the genotypic space implies small variations in the fitness value. A direct consequence is that any reliable algorithm should be able to perform both local search (to exploit this strong causality property) and global search (to avoid the pitfall of multi-

modality) ;
– Evaluation reliability : as the environmental condition vary over time depending on the robot location, performance assessment (ie. fitness) of one genome might be completely different from one starting location to another (e.g. starting in a narrow bridge or starting in the middle of an empty arena). This is the problem of noisy fitness evaluation, which requires a great number of independant evaluation to assess for the "true" performance of one genome ;

The (1+1)-online adaptation algorithm described in [3] has been shown to address these issues and provide an efficient way to perform continuous adaptation on a single e-puck robot in Player/Stage, running a Cortex M3 micro-controller. The (1+1)-online algorithm is described as a genetic algorithm based on the (1+1)ES[18], with only two genomes : a champion and a challenger, and some specific properties so as to address online adaptation :

– **Local and global search :** A mutation operator is used to produce a child from a parent. This mutation operator is able to do both local and global search. A gaussian distribution $N(0, \sigma)$ is used. The switching between local and global search is done by the value of σ. If this value is low, few modifications will be done to the genome, and the search will remain local. If the value of σ is high, the genome will be strongly modified, and the research will go global.
– **Re-evaluation :** Individuals may get lucky or unlucky during evaluation depending on the envrionment at hand. This is a typical problem related to fitness noise. An efficient solution is to reevaluate individuals, as proposed by Beyer [10].The reevaluated fitness overwrite the fitness of the champion. This is done to promote individuals with a low variance in their performances. One of the drawback of the overwriting method is that good individuals could be replaced by inferior but lucky individuals. If an individual is lucky during its first evaluation but has a low mean fitness it will not survive next-reevaluations. As a consequence, the evolutionary algorithm won't be stuck with bad individuals.
– **Recovery :** As this work assumes the evolutionary algorithm should run without human intervention, it implies no repositioning of the robot after each evaluation of one individual. For example, a genome may be evaluated starting from completely different initial conditions, such as in front of a wall or in a tight corner. To avoid penalization of good genomes, a *recovery period* is introduced : during this time, the robot behavior is not considered for evaluation (ie. "free of charge"), which favors genomes that display good performance whatever the starting position.

In this paper, we present an analysis of the global search feature of this algorithm and identify a problem that negatively impact the search. The basic idea is that the previous implementation of the (1+1)-online algorithm implies a limitation in the efficiency of global search by restraining, possibly drastically, the search space to consider. This problem is described and a new algorithm, termed (1+1)-restart-online is devised. Preliminary experiments in simulation are described and show that the new algorithm actually performs a larger global search, avoiding the pitfall of getting stuck in a local optima for a long time. Moreover, this paper describes the implementation and successful evaluation of this new algorithm on a real robotic hardware setup, a four wheels Bioloid mobile robot, in the real world.

2. EXTENDING THE (1+1)-ONLINE EA

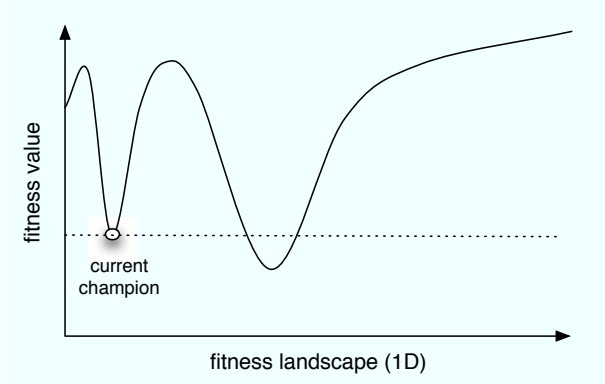
This section shed some light on an intrinsic limitation of the (1+1)-online algorithm, which possibly dramatically slow down adaptation under very specific conditions (multi-modal objective function with few or no amount of noise). Then, an extension of the previous algorithm is described that makes it possible to both retain the properties of the original algorithm as well as to address the problem identified.

2.1 Limits of the (1+1)-online

The (1+1)-online algorithm has been shown to be quite efficient in [3]. One of its main properties is to rely on a tunable gaussian mutation operator to switch between local and global search. This is achieved through a parameter, termed σ : the higher the σ, the more global the search. However, the current champion genome is replaced if and only if the challenger genome performs strictly better. While this seems to be relevant in most case, this scheme has a major drawback as it limits the search regions to be considered : only regions with better performing genomes can be considered. Figure 1 illustrates this : the fitness values of all genomes is shown (for the sake of simplicity, we assume this is a minimization task for a one dimension only problem). In this example, the current champion may be replaced *only* by a challenger which is *under* the dashed line, would it be during local or global search. In this typical setup, this may not be a relevant strategy as the probability to randomly jump to the relevant region is very low compared to the probability of picking a genome from which local search may slowly, but surely, lead to the best genome.

The modification of σ is a good candidate to find new individuals. When it is increasing the search goes more global. But at some point the search area is so constrained that it is more interesting to simply restart the whole optimization process in order to obtain an unconstrained global search. To some extent, this problem may not occur in all situations. Firstly, this problem would never occur when optimizing a convex objective function, which is unfortunately quite scarce in this setup. Secondly, very noisy objective function may cope with this problem as any good performing individual may eventually be re-evaluated with a low fitness value, and thus lead to considering the whole search space all over again – this was indeed the case in the experiments shown in [3].

Figure 1 : problematic fitness landscape (minimization task)



2.2 The (1+1)-restart-online algorithm

Escape from local minimum is a classical problem for the global search algorithms, and has been studied in different fields. A popular method is the restart strategy such as in [1]. In this setup, the algorithm is restarted, either with similar or different parameters, whenever the search is identified as stalled. This approach provides interesting results on multi-modal problems as it ensures global

convergence (ie. asymptotic exploration of the whole space is guaranteed in the worst case as restarting alone is similar to a random search).

In order to implement restart in the (1+1)-restart-online algorithm, the restart criterion has to be considered, and candidates are mostly limited to the two following :

- **Value of σ :** If σ is at its maximal value, it means that a local minimum has been reached and the search is going global. To be sure that the algorithm will never be blocked in a local minimum, it can be restarted as soon as sigma reaches its maximal value.
- **Number of champion reevaluations :** If the champion isn't replaced, it is the best in a certain area of the fitness landscape.Thus, surviving many re-evaluations assess for the robustness of the champion with regards to both other challenger genomes and to the environment. Therefore, a high number of re-evaluations can be used to detect a good performing genome, but also that search is stalled.

Using the value of σ to restart the algorithm is too constraining. There is always a probability non-equal to zero that σ reaches its maximal value without the champion being reevaluated. So, when σ is equal to its maximal value, the champion may still be unreliable. Moreover, even if this champion has been successfully reevaluated while σ was increasing, it can still be improved by mutations. On the contrary, if the champion survives many reevaluations, it is a good and reliable individual that will be hard to replace. That's why the number of reevaluation is used as a restart criterion in the restart (1+1)-online algorithm described by algorithm 1. Hence, whenever restart is triggered, the current champion is recorded in the hall-of-fame as a relevant genome and the algorithm is re-initialized with the same parameters, but from a different randomly chosen genome (uniform sampling in the genotypic space).

Algorithm 1 The restart (1+1)-ONLINE evolutionary algorithm.

```
for evaluation = 0 to N do
  if random() < P_reevaluate then
    if reevaluation_count < reevaluation_max then
      Recover(Champion)
      Fitness_Champion = RunAndEvaluate(Champion)
      reevaluation_count = reevaluation_count + 1
    else
      σ = σ_min
      Champion = RandomGenome()
      Fitness_Champion = 0
      Challenger = RandomGenome()
      Fitness_Challenger = 0
      reevaluation_count = 0
    end if
  else
    Challenger = Champion + N(0, σ) {Gaussian mutation}
    Recover(Challenger)
    Fitness_Challenger = RunAndEvaluate(Challenger)
    if Fitness_Challenger > Fitness_Champion then
      Champion = Challenger
      Fitness_Champion = Fitness_Challenger
      σ = σ_min
    else
      σ = σ · 2
    end if
  end if
end for
```

3. EXPERIMENTS AND RESULTS

In this section, an experimental setup is presented so as to evaluate the performance of the (1+1)-restart-online algorithm. Results and preliminary experiments are also described and discussed.

3.1 Hardware setup

The evaluation takes place in a setup featuring actual robotic hardware, a Cortex M3 board with 256 kb memory. The Cortex board runs a robot simulated by Symbricator3D. Symbricator3D is a robot simulator developped within the Symbrion project [1] and based on delta3d [2] (An Open Source game engine which can be used for physics-based simulations). After N time-steps, the evaluation of the current controller is complete and the controller parameters are replaced with values from a new genome, which is evaluated from the location the previous controller left it in.
Figure 2 illustrates the experimental set-up with a Cortex board connected to the computer running the simulator based on delta3d. The simulated robot is equiped with two screws and 8 distance sensors (two per side). Details of the shape of the robot can be seen in figure 3. The maze environment is shown in figure 2.

Figure 2 : The experimental setup : the Cortex M3 board connected to Symbricator3d. The numbers show the reference starting positions for validating the Hall of Fame.

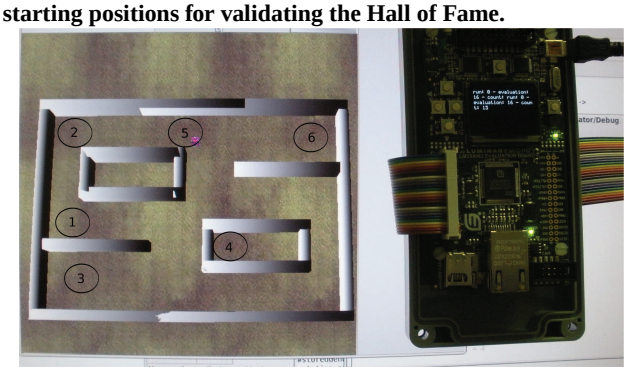
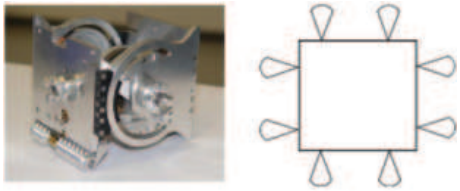


Figure 3 : Details of the Symbricator robot. (a) robot design (b) position of distance sensors (from above)



The robot is controlled by a simple perceptron with 9 input neurons (8 IR distance sensor values and one bias neuron) and 2 output neurons (translational and rotational velocities, which are combined to give actual actuator values).

3.2 Experimental Setup

The objective function used is close to the one described in [15] :

$$fitness(x) = \sum_{t=0}^{n} V_t * (1 - V_r) * i$$

1. http ://www.symbrion.eu/
2. http ://www.delta3d.org/

where V_t is the speed factor, V_r is the rotation factor, and i the value of the less active sensor, all values are normalized between 0 and 1.

The (1+1)-restart-online algorithm has been evaluated with a restart parameter fixed at 7 reevaluations. In order to compare the true performances of individuals obtained with (1+1)-online and (1+1)-restart-online, two Hall-of-Fames are computed from the results of the simulations. One containing the best individuals of (1+1)-online, the other containing the best individuals of restart (1+1)-online. The value for each individual in the Hall-of-Fame corresponds to the sum of the re-evaluated fitness obtained by this individual during the experiments.

While the adaptation process could go on forever, an arbitrary number of evaluations is fixed by the supervisor. Afterwards, an experimental protocol is used to compare the best individuals from the Hall-of-Fame : every individuals from the Hall-of-Fames are evaluated from 6 reference starting positions shown in 2 during 120 time step [3]. This validation protocol provides fair comparaison between genomes.

3.3 Experimental Results

Figure 4 shows evolution dynamics of a critical run of the (1+1)-online algorithm. Evaluations are denoted on the x-axis. The y-axis is divided in two parts. The top half shows the fitness of the current champion in green dashed line. The bottom half shows the number of re-evaluations of the current champion (downwards ; the lower the line, the higher the number of re-evaluations). The small vertical markers near the x-axis indicate whenever a new champion is adopted, i.e., when the challenger outperforms the current champion. During this run a good champion has been found at evaluation 180, and hasn't been replaced until evaluation 538 after 64 reevaluation. This problem is detected in our work and not in [3] because the robot and the simulator are different, especially with regards to noise between evaluations. This is a typical illustration of the problem identified in this paper with the original (1+1)-online algorithm : a less noisy setup is prooved to be more deceitful for the original algorithm.

Figure 5 shows of a run of the (1+1)-restart-online algorithm. In this run one can think that the algorithm was restarted around evaluation 132, but in this case it is the reevaluation procedure that take place. Indeed, a lucky individual has been found at evaluation 126, and has been reevaluated at evaluation 132. This shows that the reevaluation mechanism is still useful to detect lucky individuals. In this run the restart procedure is used at evaluation 368, to replace a robust champion. According to preliminary experiments, it seems that the champion of evaluation 368 could still be improved, which may imply that the restart strategy could be triggered later.

3.4 Hall-of-Fame analysis

As described in section IV.B, two Hall-of-Fames were maitened during the course of the experiments, with each Hall-of-Game computed out of 14 independant runs of 600 evaluations. There are 1691 individuals in the Hall-of-Fame obtained by running the (1+1)-online algorithm, and 2158 individuals in the Hall-of-Fame obtained by running the (1+1)-restart-online algorithm. This difference is a desired effect of the (1+1)-restart-online algorithm as the restart feature favors exploration by saving unnecessary reevaluations

3. The starting position number 4 is an extreme case where the robot is tested in a hard environment never seen before.

Figure 4 : Evolution dynamics of a critical run of the (1+1)-online algorithm

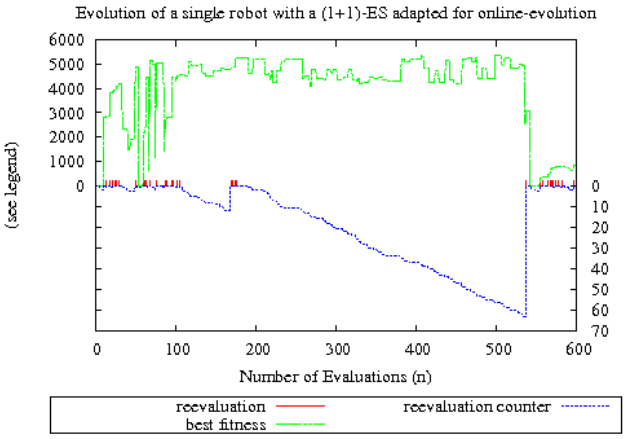
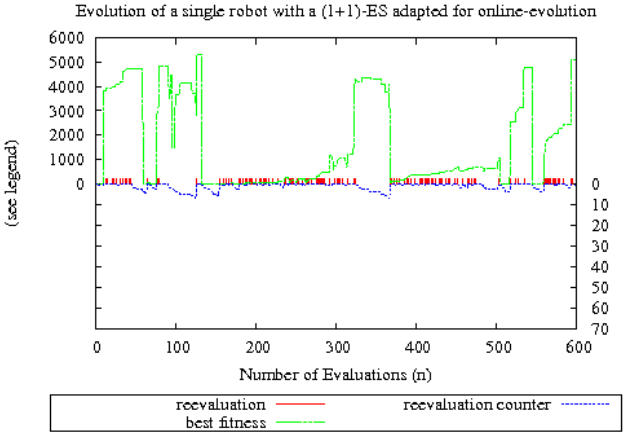


Figure 5 : Evolution dynamics of a run of the (1+1)-restart-online algorithm

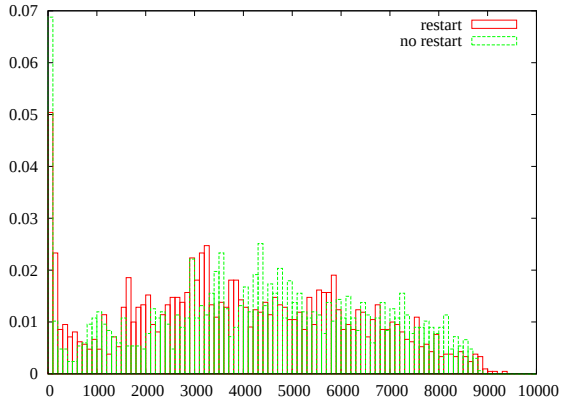


of champions whenever the algorithm is stalled.

Then, performances of the best individuals generated by the (1+1)-restart-online algorithm and by the (1+1)-online algorithm are compared. As described in section IV.B, every individuals from the Hall-of-Fames are evaluated from six pre-defined positions, so as to provide comparable figures. For each individual the mean performance obtained from those 6 positions has been computed. The figure 6 displays the fitness density for this validation. The x-axis shows the different fitness obtained during the validation of the 628 best individuals of each Hall-of-Fame. The y-axis shows the number of individuals with the same fitness. It is clear that there is no loss of efficiency with the (1+1)-restart-online algorithm.

From the two previous considerations, it should be noted that while the number of evaluations in the experiments shown here provides enough individuals to get similar results, the (1+1)-restart-online algorithm is faster - which is a key feature to provide many candidates whenever ressources are limited.

Figure 6 : Fitness density of the best individuals produced by the (1+1)-online algorithm, and the (1+1)-restart-online algorithm



3.5 Real robot experiment

The (1+1)-restart-online has been tested on an autonomous four-wheels Bioloid robot. The Bioloid kit provides robotic parts and an ATmega128 microcontroller with 128Kb of memory. Figure 7 shows the robot used in this work. It is equiped with 4 motors, and 7 distance sensors. The 7 red arrows in figure 7 shows the orientations of the distance sensors. The controller of the robot is a feedforward neural network with 8 inputs (7 distance sensors and 1 bias) and 2 outputs (left and right motor velocities). The two left side wheel velocities are controled by the same neuron, and the two right wheel velocities by the other one. The fitness function used is the same as the one described in section 3.2. Each individual is recovering during 60 time steps (7 seconds) and is evaluated during 60 following time steps (7 seconds). As in section 3.2 the restart threshold is fixed to 7 re-evaluations. The experiment lasted 1 hour and 10 minutes during which the robot was completely autonomous.

Figure 7 : (a) The robot and the directions of the 7 distance sensors, (b) the environment

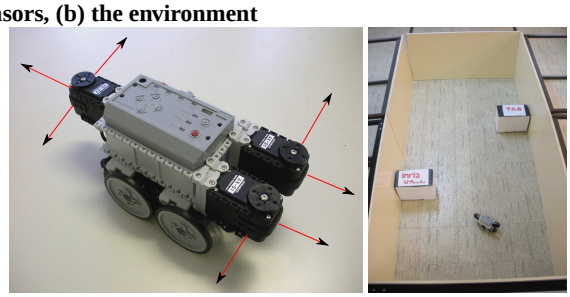


Figure 7 (b) shows the experimental setup.

The algorithm provides similar figures to what has been already shown in the previous experiment and the traces of the first two best evolved controllers from the Hall-of-Fame are illustrated in figures 8 and 9. These two control architectures efficiently avoid wall with simple yet efficient behaviors. The best controller (figure 8) is faster when moving in straight line, and displays sharper turn trajectories. Other genomes have been empirically evaluated (not shown here) and display mostly the same kind of behaviors as these two, but with minor differences (sharper turn, slower/faster trajectories, etc.).

Figure 8 : Example of behavior for the best evolved controller.

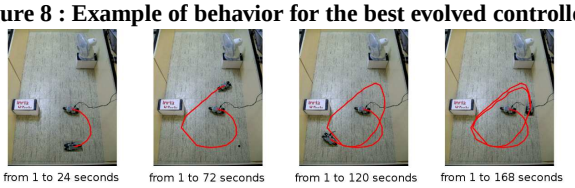
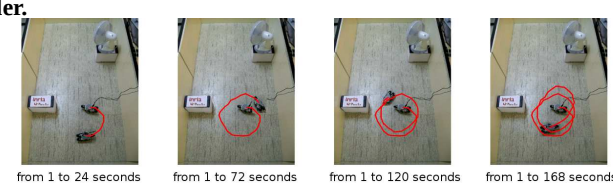


Figure 9 : Example of behavior for the 2nd best evolved controller.



An interesting remark about this experiment is that relying the online adaptation algorithm considered makes it straight-forward to address an important issue in Evolutionary Robotics, that of the reality gap[12]. Indeed, the algorithm needed exactly the same amount of work from the experimenter in simulation and reality and *neither* human intervention *nor* external remote control was ever needed during the whole experiment with the real robot. Of course, this assumption must be taken with care as the fitness considered here is a rather simple one and was chosen so that it was possible to focus on the validation of the algorithm features rather than on the algorithm's ability to solve a complex problem.

4. CONCLUSION AND PERSPECTIVES

In this paper, the problem of online onboard behavior adaptation for a single autonomous mobile robot has been adressed. Precisely, the (1+1)-online adaptation algorithm from [3] is studied and a limitation of this algorithm is identified and analysed regarding its ability to perform global search in the space of possible solutions. A new algorithm is described, termed (1+1)-restart-online, and was shown to efficiently address the trade-off between local and global search by relying on a restart procedure whenever the algorithm is stuck in a local optima. This restart procedure makes it possible to address a previous design flaw by relaxing some constraint over the search space to be considered.

This algorithm has been evaluated both within a real micro-controller connected to a Symbricator Robot running in simulation and within a real mobile robot with four wheels and 7 proximity sensors. Results have shown that this new algorithm is actually able to provide wider exploration of the search space, potentially making it possible to visit many more local optima that the previous implementation, and possibly increasing the probability to end up in a global optima. Moreover, this algorithm has been shown to be straight-forwardly use within a real robot platform in a complete autonomous fashion, which makes it possible to naturally address the reality gap issue.

On-going work focus on evaluating precisely the advantages of this new algorithmn in particular focusing on the distribution of the performance from all individuals in the Hall-of-Fame. Moreover, the

new restart feature in the algorithm is being carefully studied as there exists a possible trade-off in balancing the previous global search strategy and the new restart strategy. Indeed, choosing between the two strategies clearly depends on both the shape of the fitness landscape and actual local minimum as this trade-off can be reformulated as favoring global search over avoiding re-convergence towards already visited local optima.

Future works will address the problem of noisy fitness evaluation by extending the $(1 + 1)$ stragegy into a $(\mu + 1)$ strategy, which roughly means that a reservoir, or a distribution, of champion genomes will be considered rather than only a single champion genome. Also, the extension towards multi-robots is rather straight forward as one can consider the current adaptation algorithm to act within one island of a distributed evolutionary algorithm. In this setup, each robot/island runs an embedded adaptation algorithm, where best genomes may be exchanged from one island to another, as in the well-known GA island model[21].

5. ACKNOWLEDGMENTS

This work was made possible by the European Union FET Proactive Intiative : Pervasive Adaptation funding the Symbrion project under grant agreement 216342 (Symbrion EU Project).

6. REFERENCES

[1] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005.*, 2005.

[2] V. Braintenberg. *Vehicles : Experiments in Synthetic Psychology*. MIT Press, 1986.

[3] N. Bredeche, E. Haasdijk, and A. E. Eiben. On-line, on-board evolution of robot controllers. *Submitted*, 2009.

[4] A. Eiben and Z. Michalewicz, editors. *Evolutionary Computation*. IOS Press, 1998.

[5] S. Elfwing. *Embodied Evolution of Learning Ability*. PhD thesis, KTH School of Computer Science and Communication, SE-100 44 Stockholm, Sweden, November 2007.

[6] S. Ficici, R. Watson, and J. Pollack. Embodied evolution : A response to challenges in evolutionary robotics. In J. L. Wyatt and J. Demiris, editors, *Proceedings of the Eighth European Workshop on Learning Robots*, pages 14–22, 1999.

[7] D. Floreano, P. Husbands, and S. Nolfi. Evolutionary robotics. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, pages 1423–1451. Springer, 2008.

[8] D. Floreano and C. Mattiussi. *Bio-Inspired Artificial Intelligence : Theories, Methods, and Technologies*. Intelligent Robotics and Autonomous Agents. MIT Press, Cambridge, MA, 2008.

[9] D. Floreano, N. Schoeni, G. Caprari, and J. Blynel. Evolutionary Bits'n'Spikes. In R. K. Standish, M. A. Beadau, and H. A. Abbass, editors, *8th International Conference on the Simulation and Synthesis of Living Systems (Alife 8)*. MIT Press, 2002. In R. K. Standish, M. A. Beadau and H. A. Abbass (eds).

[10] H. georg Beyer. Evolutionary algorithms in noisy environments : Theoretical issues and guidelines for practice. In *Computer Methods in Applied Mechanics and Engineering*, pages 239–267, 1998.

[11] S. Haroun Mahdavi and P. J. Bentley. Innately adaptive robotics through embodied evolution. *Auton. Robots*, 20(2) :149–163, 2006.

[12] N. Jakobi, P. Husband, and I. Harvey. Noise and the reality gap : The use of simulation in evolutionary robotics. In F. Moran, A. Moreno, J. Merelo, and P. Chancon, editors, *Advances in Artificial Life : Proceedings of the Third European Conference on Artificial Life*. Berlin : Springer Verlag, 1995.

[13] L. Koenig, K. Jebens, S. Kernbach, and P. Levi. Stability of on-line and on-board evolving of adaptive collective behavior. In *European Robotics Symposium 2008*, volume 44/2008 of *Springer Tracts in Advanced Robotics*, pages 293–302. Springer Berlin / Heidelberg, MAR 2008.

[14] U. Nehmzow. Physically embedded genetic algorithm learning in multi-robot scenarios : The pega algorithm. In C. Prince, Y. Demiris, Y. Marom, H. Kozima, and C. Balkenius, editors, *Proceedings of The Second International Workshop on Epigenetic Robotics : Modeling Cognitive Development in Robotic Systems*, number 94 in Lund University Cognitive Studies, Edinburgh, UK, August 2002. LUCS.

[15] S. Nolfi and D. Floreano. *Evolutionary Robotics : The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA : MIT Press/Bradford Books, 2000.

[16] A. L. F. Perez, G. Bittencourt, and M. Roisenberg. Embodied evolution with a new genetic programming variation algorithm. *icas*, 0 :118–123, 2008.

[17] I. Rechenberg. *Evolutionstrategie : Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, 1973.

[18] H.-P. Schwefel. *Numerical Optimisation of Computer Models*. Wiley, New York, 1981.

[19] E. D. V. Simões and K. R. Dimond. Embedding a distributed evolutionary system into population of autonomous mobile robots. In *Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference*, 2001.

[20] S. Thrun, W. Burgard, , and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[21] M. Tomassini. Spatially structured evolutionary algorithms : Artificial evolution in space and time. In *Natural Computing Series*, 2005.

[22] Y. Usui and T. Arita. Situated and embodied evolution in collective evolutionary robotics. In *Proceedings of the 8th International Symposium on Artificial Life and Robotics*, pages 212–215, 2003.

[23] J. H. Walker, S. M. Garrett, and M. S. Wilson. The balance between initial training and lifelong adaptation in evolving robot controllers. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 36(2) :423–432, 2006.

[24] R. A. Watson, S. G. Ficici, and J. B. Pollack. Embodied evolution : Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1) :1–18, April 2002.

[25] S. Wischmann, K. Stamm, and F. Wörgötter. Embodied evolution and learning : The neglected timing of maturation. In F. Almeida e Costa, editor, *Advances in Artificial Life : 9th European Conference on ArtificialLife*, volume 4648 of *Lecture Notes in Artificial Intelligence*, pages 284–293. Springer-Verlag, Lisbon, Portugal, September 10–14 2007.